Introduction to 3D Deep Learning

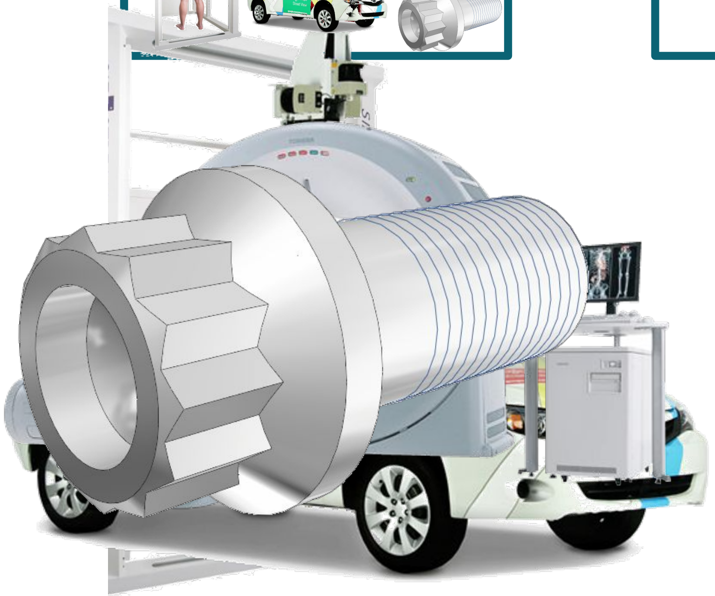
Jun. 13 2021

Slides made by Rana Hanocka

Motivation

Geometry Processing

Acquisition



Processing

Applications

Geometry Processing

Acquisition

Processing

Applications

Robotics

Geometry Processing

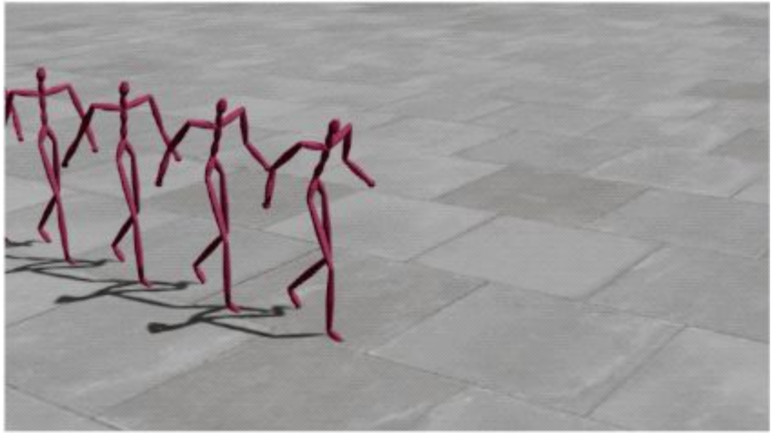
Acquisition

Processing

Applications

Robotics

Animation



Aberman et al. [2020]

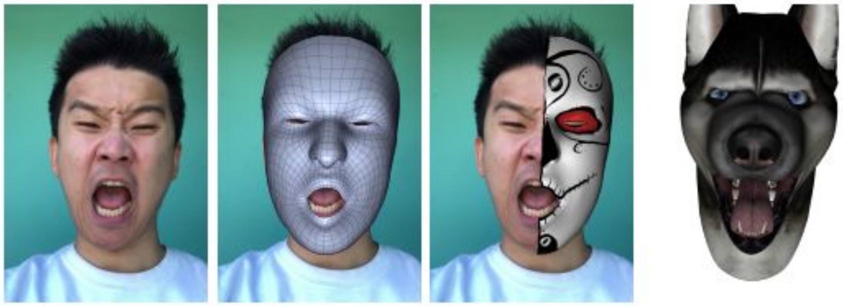
Geometry Processing

Acquisition

Processing

Applications

Robotics
Animation
VR/AR



Cao et al. [2018]

Geometry Processing

Acquisition

Processing

Applications

Robotics
Animation
VR/AR
Manufacturing
Engineering
Architecture
Biomedical
Computer Vision
…

Better Geometry Processing Equals...

Acquisition

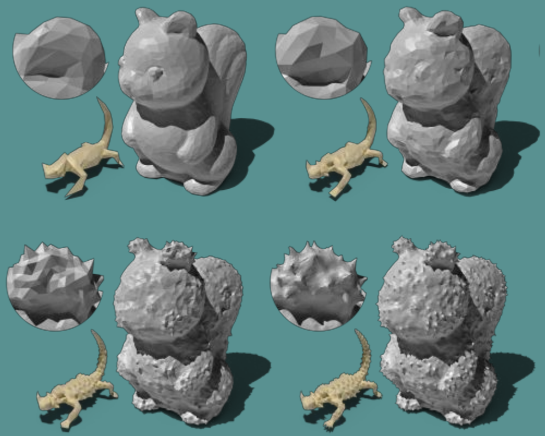
Processing

Applications

3D Deep Learning

Easier real-world application

More geometric data in the wild



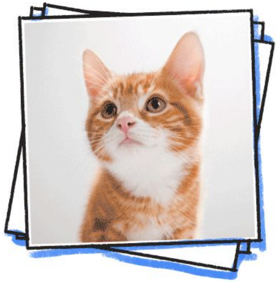
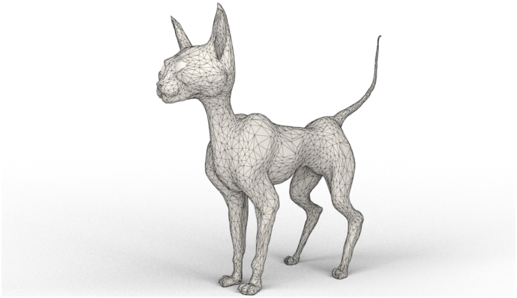
Teasers

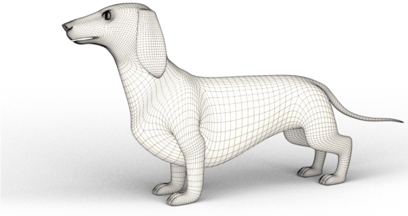
Quick overview of some 3D DL progress on classic geometry processing problems

Many similar tasks to images

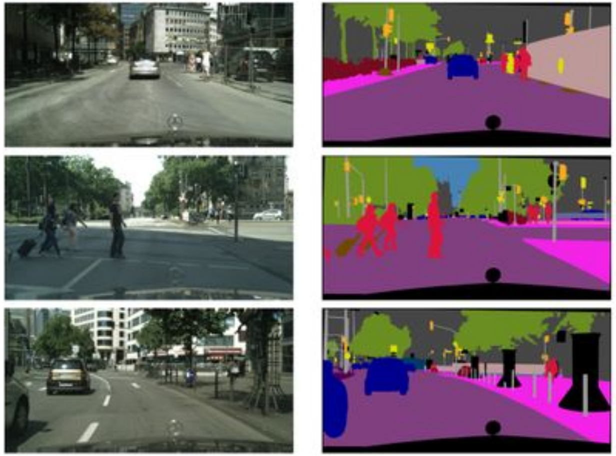
Basic ones:

- Classifying images -> classifying shapes

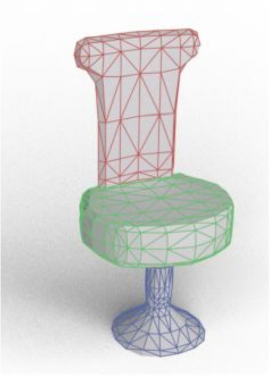


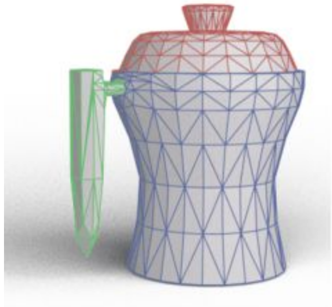
Many similar tasks to images

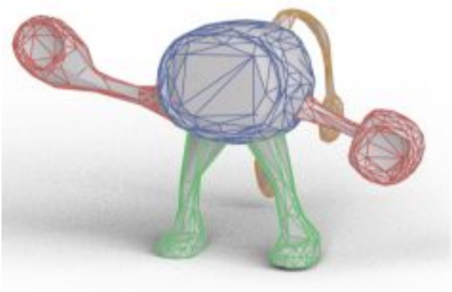


Basic ones:

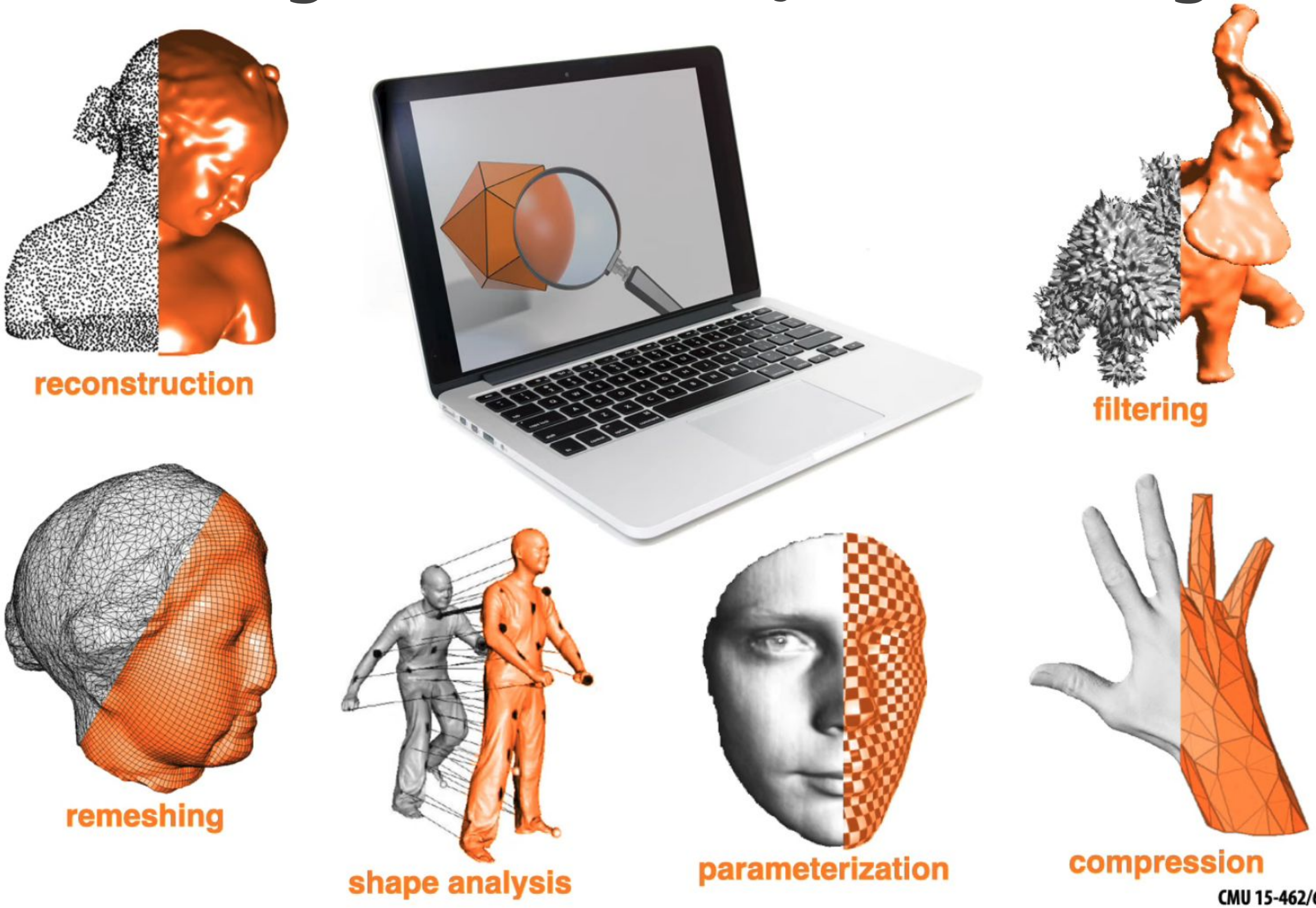
- Classifying images -> classifying shapes
- Segmenting images -> segmenting shapes



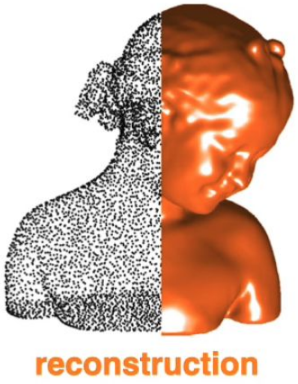




Examples of Digital Geometry Processing



Surface Reconstruction

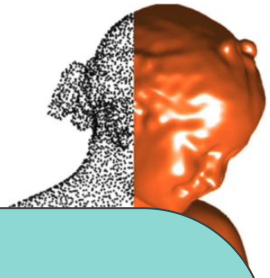


Point Cloud

Surface

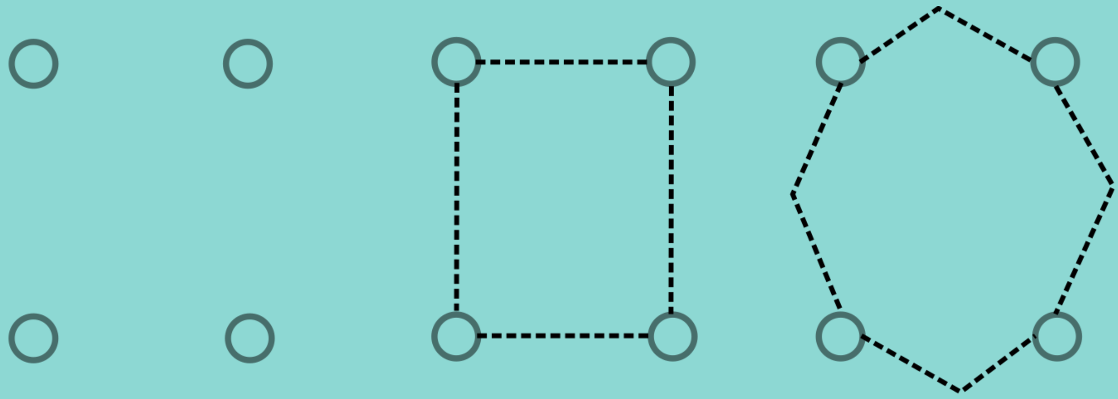
Point2Mesh: A Self-Prior for Deformable Meshes [SIGGRAPH 2020]

Surface Reconstruction



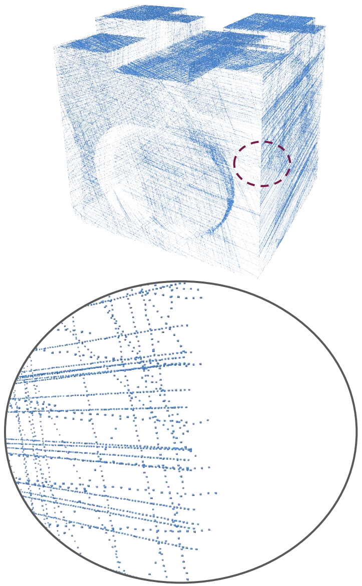
Ill-posed problem

Many solutions!

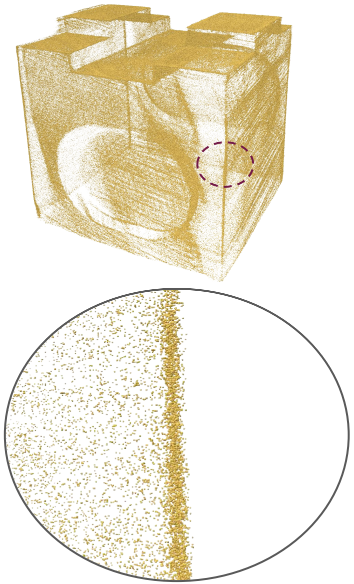


Point2Mesh: A Self-Prior for Deformable Meshes [SIGGRAPH 2020]

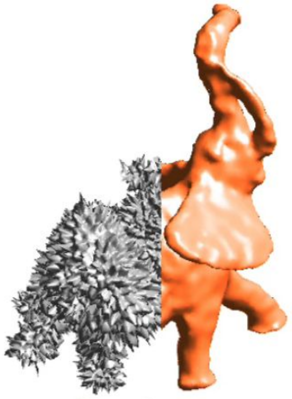
Denoising / Filtering / Consolidation



Input Scan

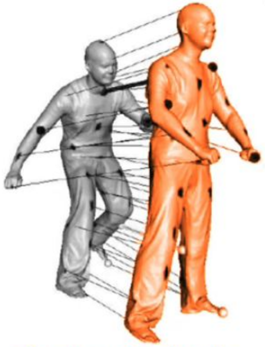


Consolidated



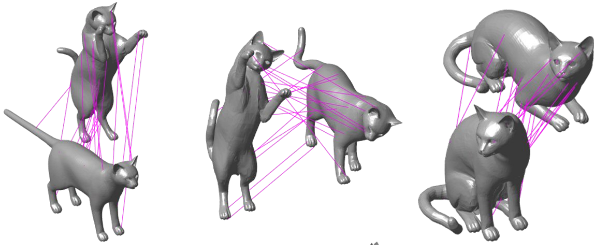
Self-Sampling for Neural Point Cloud Consolidation [TOG 2021]

Shape Matching



shape analysis

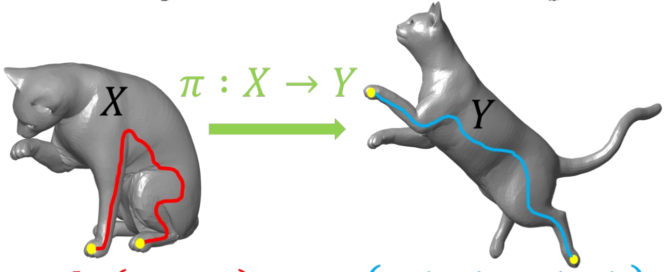
Ground Truth Knowledge



is replaced by

Geometric Invariants

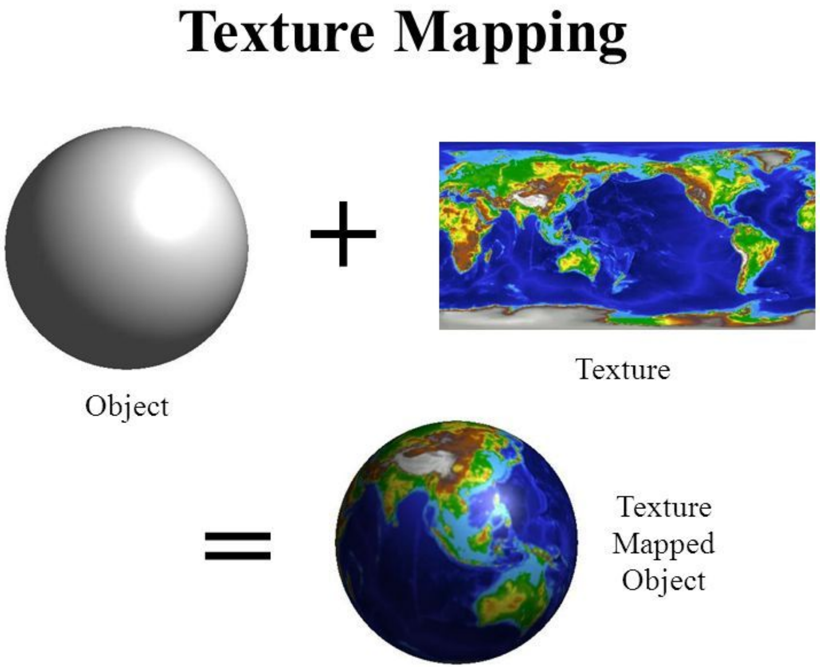
$\pi : X \rightarrow Y$



$$d_X(x_1, x_2) = d_Y(\pi(x_1), \pi(x_2))$$

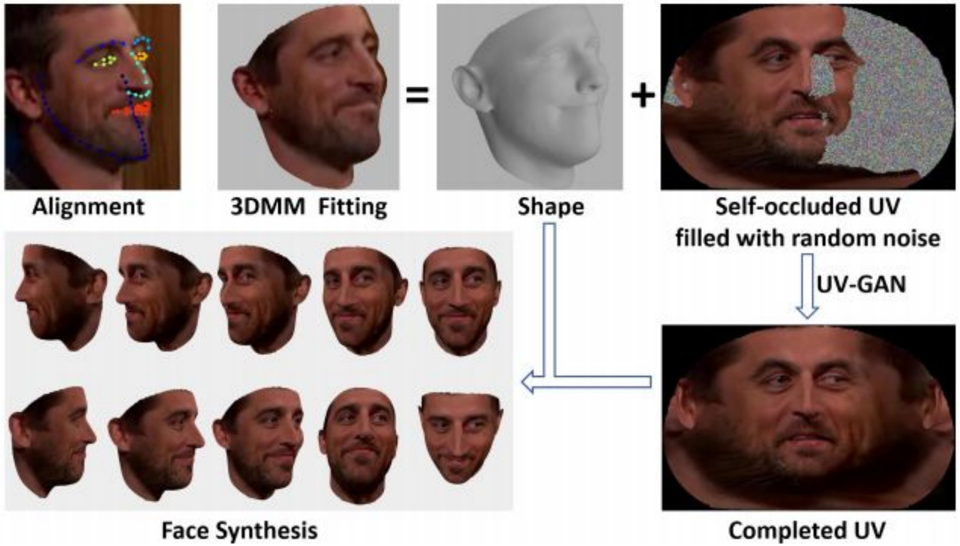
Unsupervised Learning Of Dense Shape Correspondence [CVPR 2019]

UV-Mapping

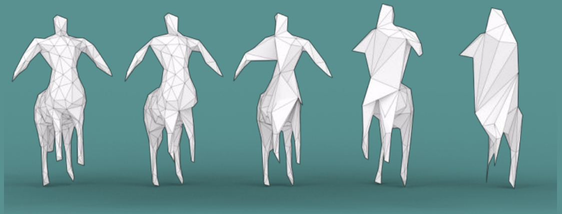

Texture Mapping
Object
Texture
Texture
Mapped
Object


parameterization

UV-Mapping

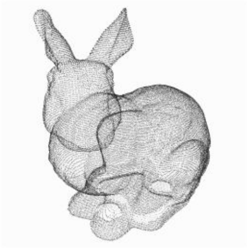


UV-GAN [CVPR 2018]

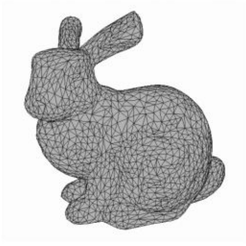


3D Shapes

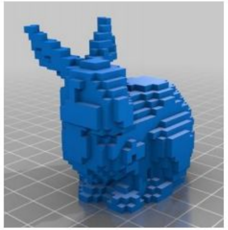
How to represent a shape?



Point Cloud

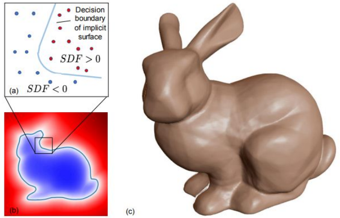


Surface Mesh



Volumetric

Multi-View
Images
RGB(D)



Implicit

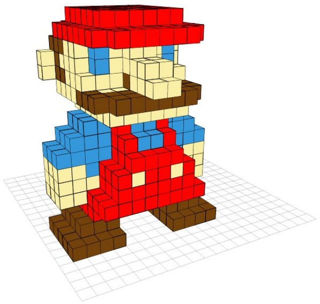
Volumetric

64x64x64 grid

Occupancy Values (usually 0 or 1)

A lot of empty cells!

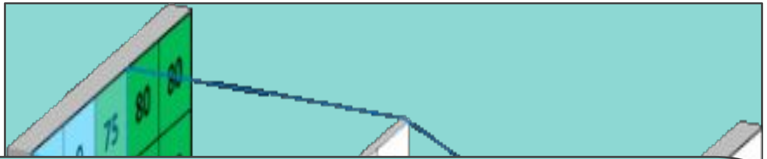
Consumes a lot of memory



First Voxel CNN:

Shapenets [Wu et al. 2015]

Volumetric CNN



Pros:

- Regular (easy application of standard operators convolution, pooling, etc.)
- Simple / flexible representation

Cons:

- Wasteful
- A lot of empty convolutions / space
- Requires a lot of GPU Memory
- Low resolution

Improve performance -> Octree-based

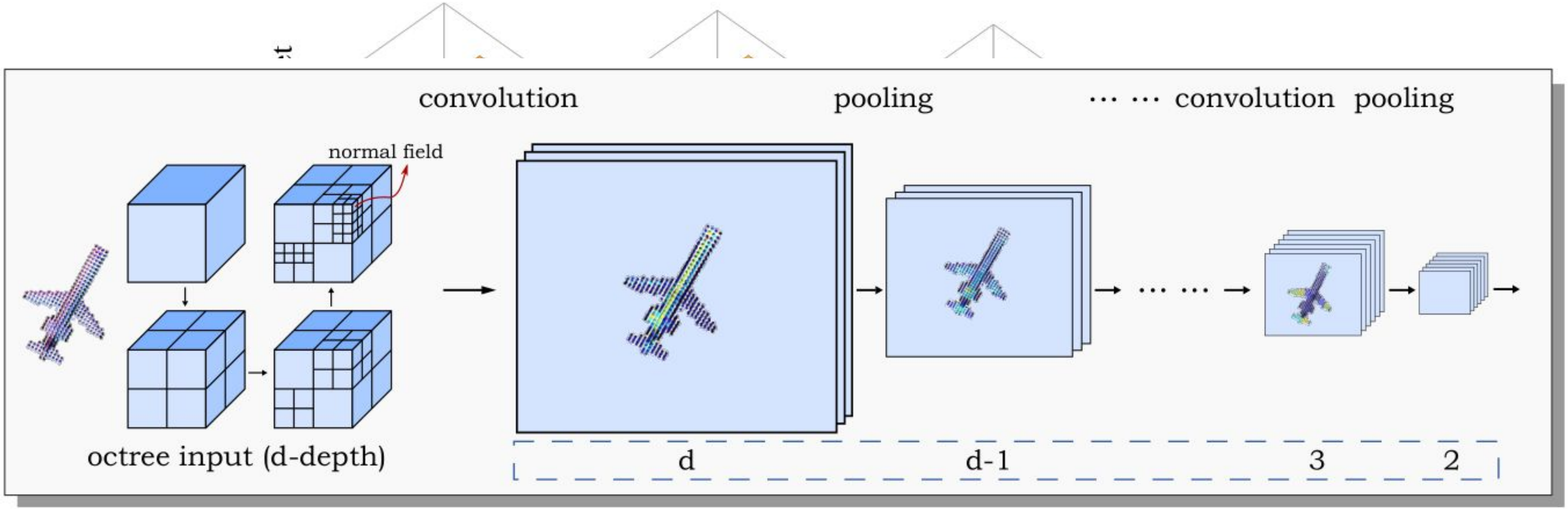
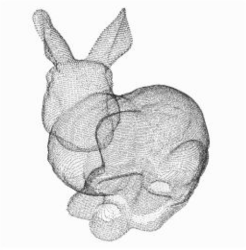
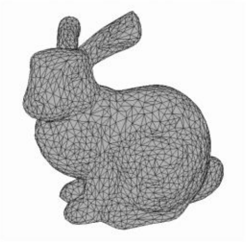


Fig. 1. An illustration of our octree-based convolutional neural network (O-CNN). Our method represents the input shape with an octree and feeds the averaged normal vectors stored in the finest leaf octants to the CNN as input. All the CNN operations are efficiently executed on the GPU and the resulting features are stored in the octree structure. Numbers inside the blue dashed square denote the depth of the octants involved in computation.

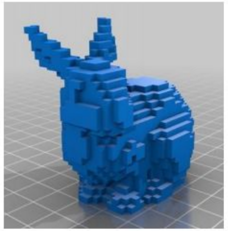
How to represent a shape?



Point Cloud



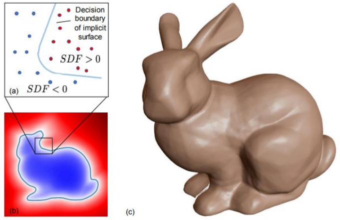
Surface Mesh



Volumetric

Multi-View Images

RGB(D)

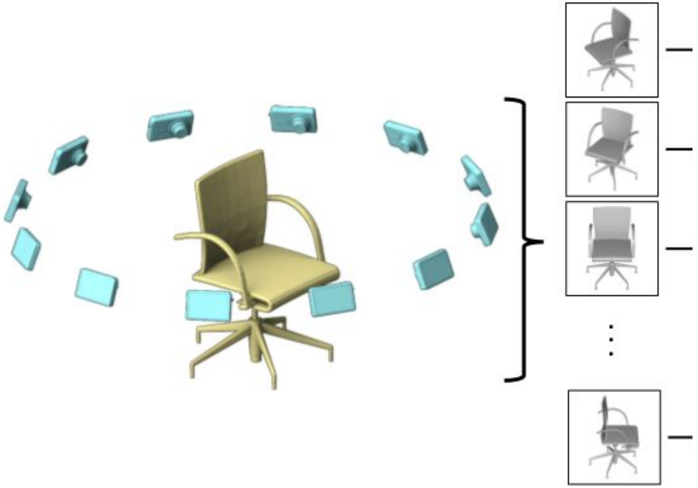


Implicit

2D Projections

- Take many 2D images of a shape

- All images together form the representation of the shape



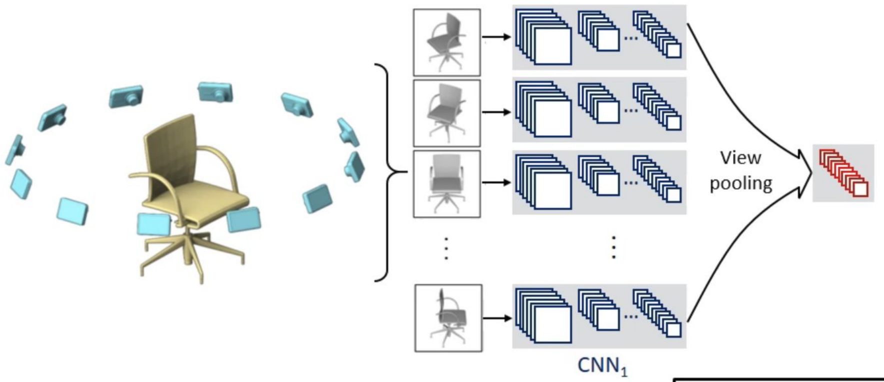
First:

MVCNN Su at al. 2015

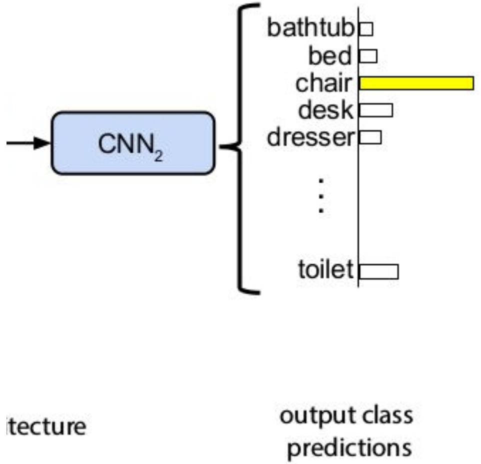
2D Projections CNN

- Each image passed through identical CNN
- Element-wise max-pooling across all views

All image features are combined by view pooling ...



View pooling: element-wise max-pooling across all views



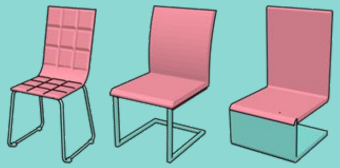
2D Projections CNN

Pros:

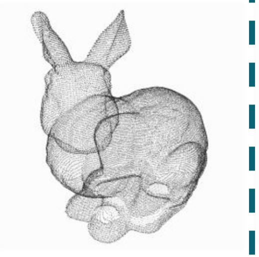
- Regular (easy application of standard operators convolution, pooling, etc.)
- Simple / flexible representation
- Leverages power of 2D CNNs

Cons:

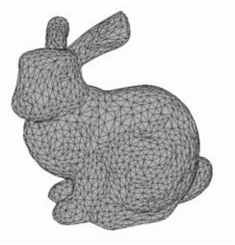
- How to determine the correct viewpoint / number of viewpoints?
- Non-trivial: generation tasks
- Non-trivial: segmentation
- Wasteful / redundant



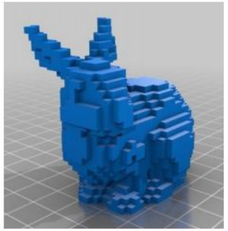
How to represent a shape?



Point Cloud



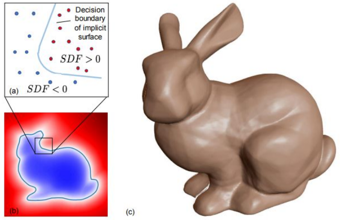
Surface Mesh



Volumetric

Multi-View Images

RGB(D)



Implicit

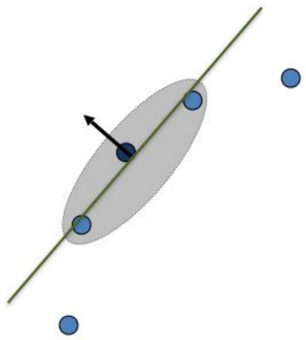
Point Cloud

- Unordered list of XYZ coordinates in space
- List of size Nx3
- Floating point values!
- Irregular
- Permutation invariant

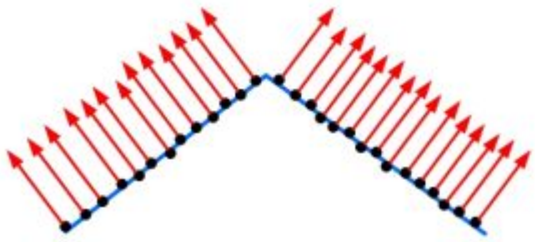
5.432	2.123	-1.23
8.14	9.42	-3.71
6.908	9.082	-2.39
3.789	4.32	-2.389

Point Cloud Normals

- Sometimes we will also use point normals as input
- Then the list is Nx6

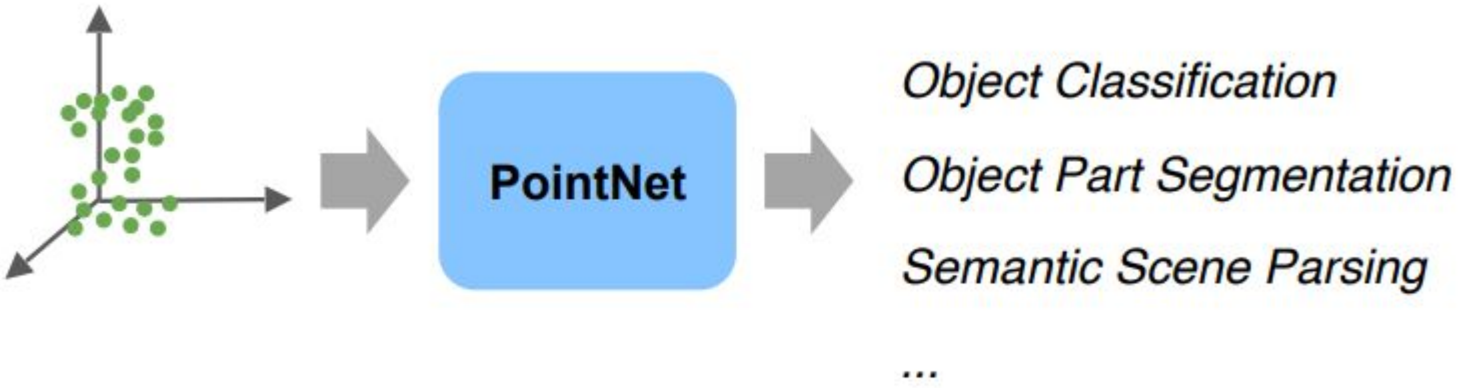


Estimate by plane fitting



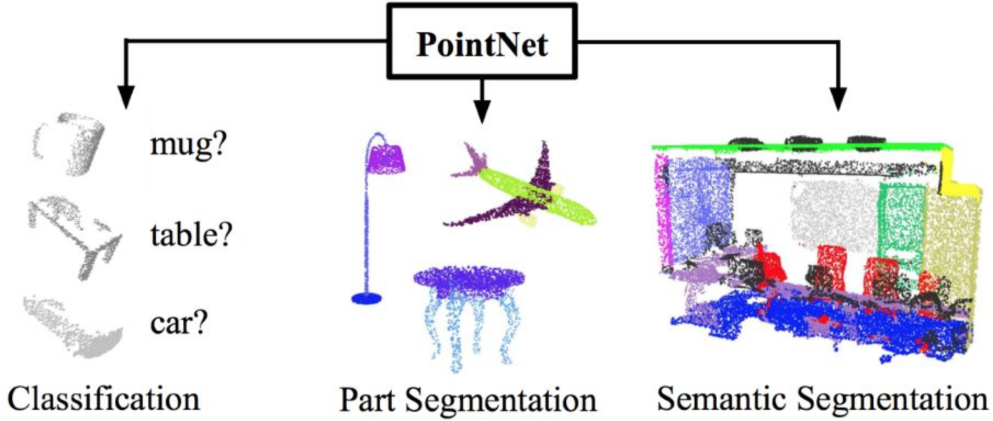
Pointnet

- Can be used for different tasks
- Generic



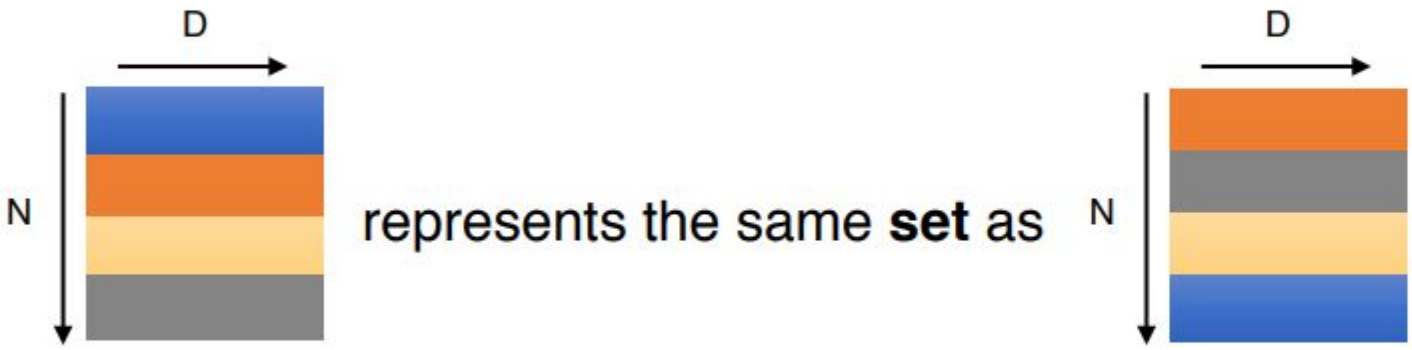
Pointnet

- End-to-end learning for scattered, unordered point data
- Unified framework for various tasks

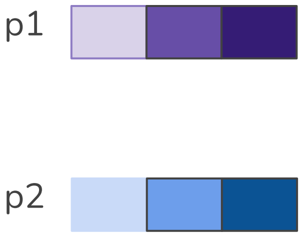


Pointnet: challenges

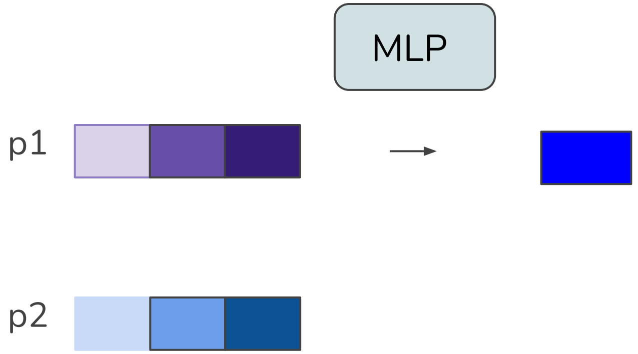
- Unordered
- Irregular



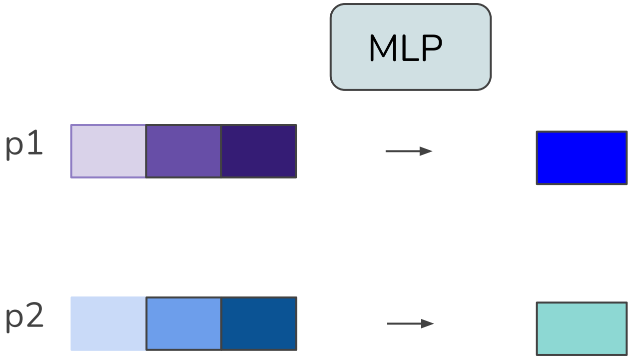
Pointnet



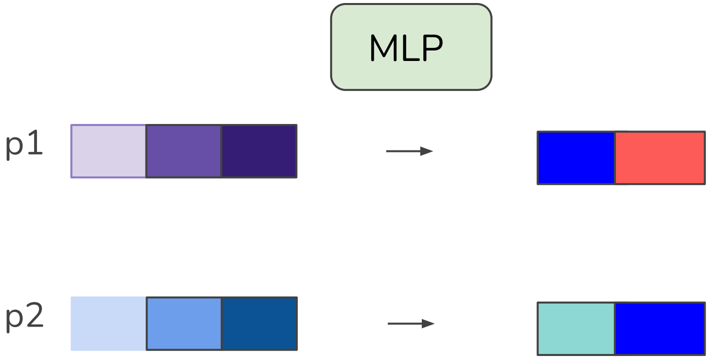
Pointnet



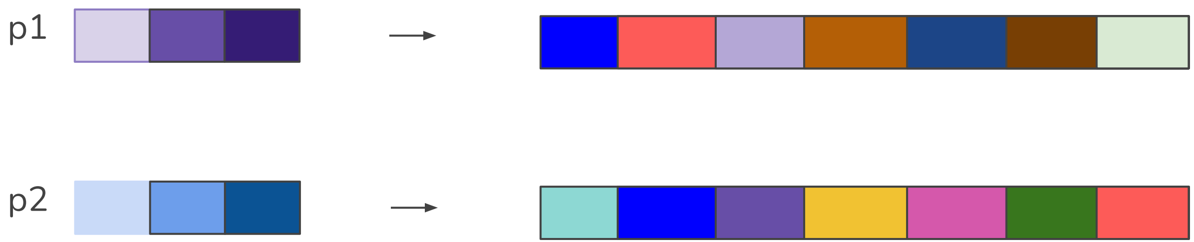
Pointnet



Pointnet

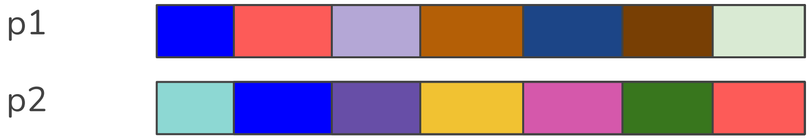


Pointnet



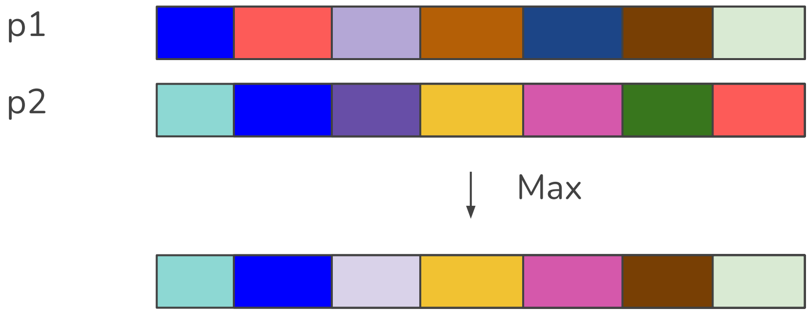
Pointnet

- Shared MLP (1x1 convolution)
- Same h applied to each point
    - Abstracted to a deep feature
- Nx3 -> Nx64 -> Nx1024 -> 1024

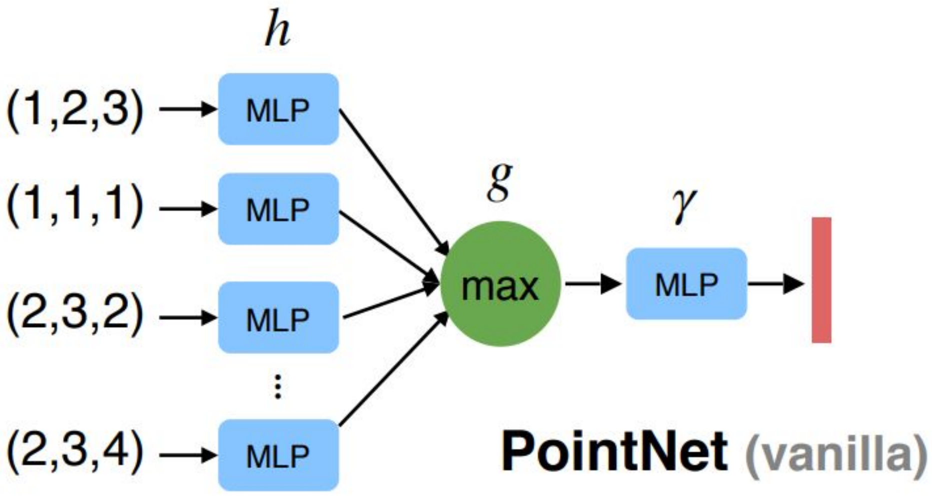


Pointnet

- Shared MLP (1x1 convolution)
- Same h applied to each point
  - Abstracted to a deep feature
- Nx3 -> Nx64 -> Nx1024 -> 1024



Pointnet



h
(1,2,3)
(1,1,1)
(2,3,2)
(2,3,4)
MLP
MLP
MLP
MLP
g
max
γ
MLP
PointNet (vanilla)

Improvements to Point Networks

Point Networks

PointNet Qi et al. [2017]

PointNet++ Qi et al. [2017]

DGCNN Wang et al. [2018]

PointCNN Li et al. [2018]

Point Networks

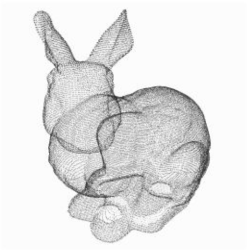
Pros:

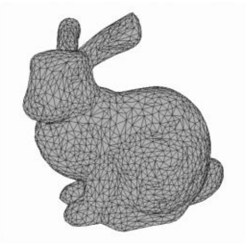
- Point clouds come directly from scanners
- Simple / flexible representation

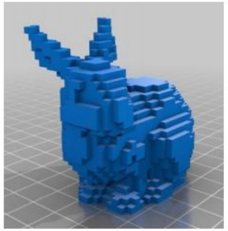
Cons:

- Don’t contain underlying shape surface / structure
- Inefficient (requires a lot of points to understand surface)

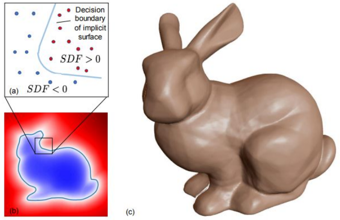
How to represent a shape?


Point Cloud


Surface Mesh


Volumetric

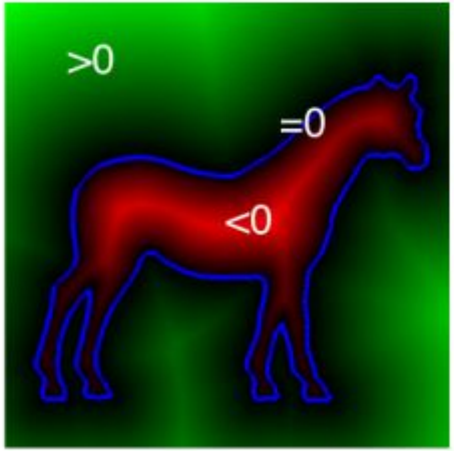

Multi-View
Images
RGB(D)



Implicit

Implicits: Signed distance function

- Each point in space is represented by a distance
- Points inside the surface have negative distance
- Points outside the surface have positive distance

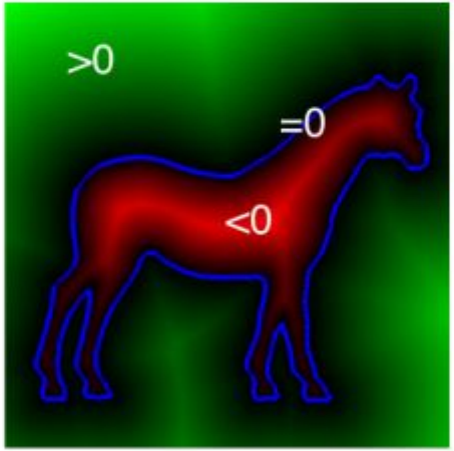


So what's a signed-distance function (SDF):

SDF(x, y, z) = d

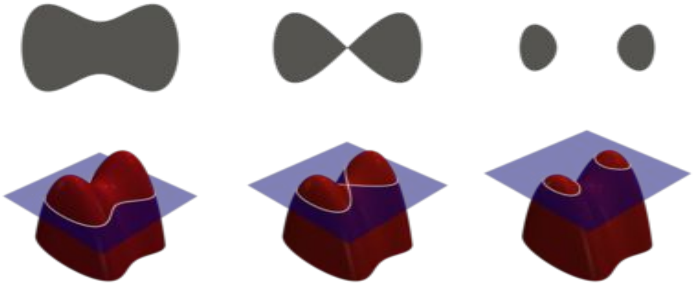
Why is it implicit?

- Doesn’t *explicitly* give us the surface!
- The surface is defined IMPLICITLY, based on the zero-crossings of the SDF



How to get explicit surface representation?

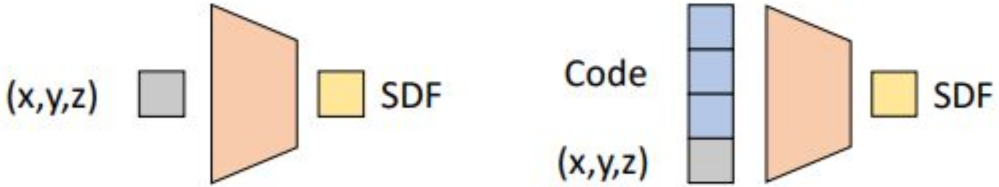
- Look for the zero-crossings or boundaries. Also called “zero level set”:



Implicit Networks

- Usually used for shape generation, as a set of point clouds with distances

DeepSDF



(a) Single Shape DeepSDF **(b)** Coded Shape DeepSDF

Figure 3: DeepSDF network outputs SDF value at a 3D query location. While (a) the network can memorize a single shape, (b) conditioning the network with a code vector allows DeepSDF to model a large space of shapes, where the shape information is contained in the code vector that is concatenated with the query point.

Implicit Networks

Pros:

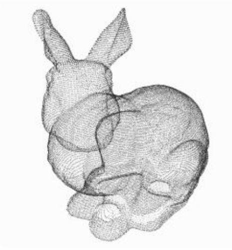
- Handles changes in genus well

Generative Adversarial Networks and Autoencoders for 3D Shapes

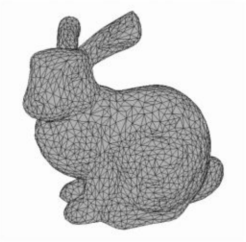
Cons:

- Unclear how to use as an encoder
- Not efficient -- requires many points to represent the surface
- Can become complicated for messy geometric data

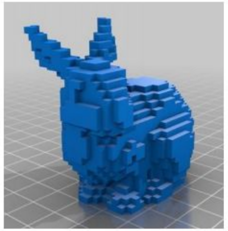
How to represent a shape?



Point Cloud

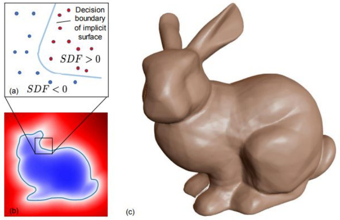


Surface Mesh

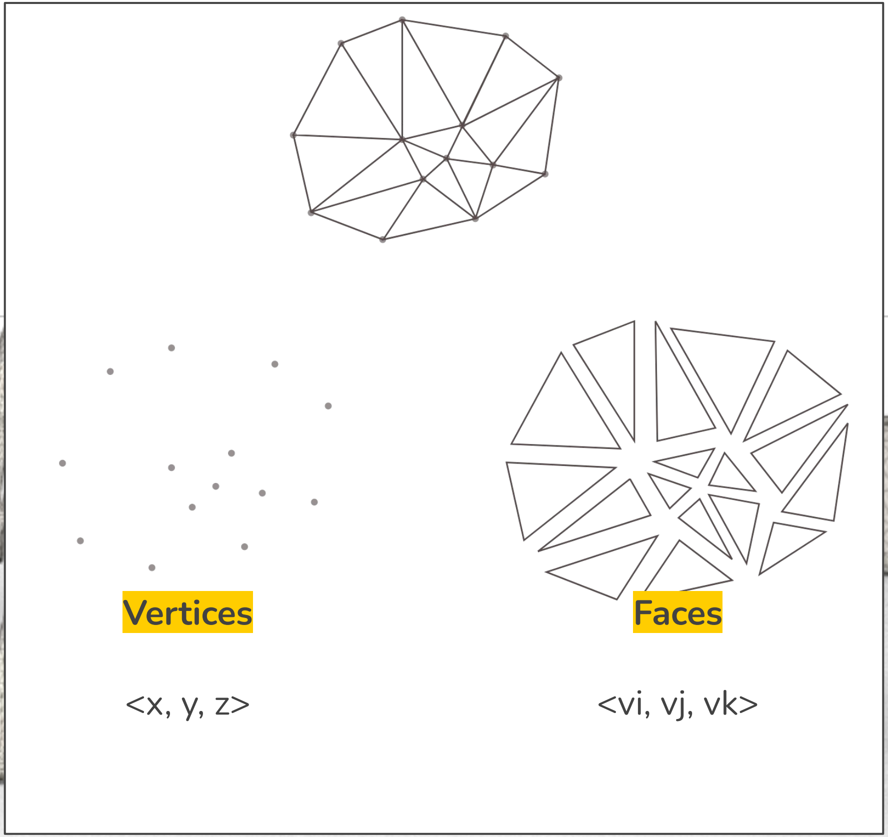


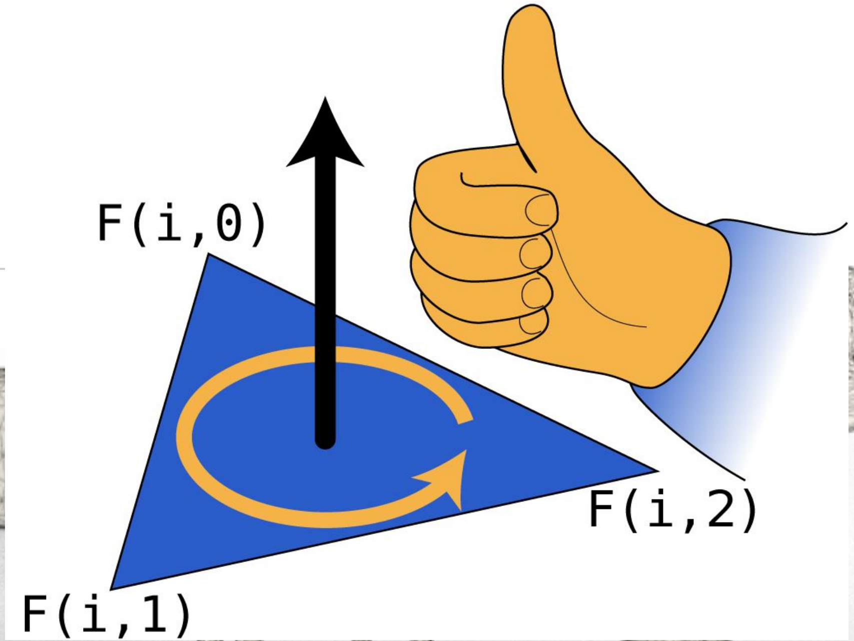
Volumetric

Multi-View
Images
RGB(D)



Implicit





Triangular Mesh

Mesh Representation

- Popular in Computer Graphics
- Non-Uniform & efficient
- Accurately portray shape
- Contains topology
- Irregular & Unordered